Applicability of NoSQL Databases to Mobile Networks: Case Home Location Register

Rasmus Paivarinta[1] and Yrjo Raivio[2]

[1] Ixonos Plc,
P.O.Box 284, FI-00811 Helsinki, Finland
rasmus.paivarinta@ixonos.com
http://www.ixonos.com
[2] Aalto University, School of Science
P.O.Box 15400, FI-00076 AALTO, Finland
yrjo.raivio@aalto.fi
http://csi.aalto.fi

Abstract. Cloud computing is slowly reaching mobile networks. Experts have doubts that cloud technologies can guarantee carrier grade service level, but the situation is rapidly changing. First of all, Infrastructure as a Service (IaaS) offers a complete computation platform, where instances can be virtually hosted either locally, remotely or in a hybrid fashion. Secondly, NoSQL (Not only SQL) databases are widely used in internet services, such as Amazon and Google, but they have not yet been applied to telecom applications. This paper evaluates, whether cloud technologies can meet the carrier grade requirements. The home location register (HLR) is benchmarked using IaaS cloud computing platforms and the HBase NoSQL database system. The main measurements are carried out in the Amazon Elastic Compute Cloud (EC2). The discussion section evaluates and compares the results with other similar research. Finally, the conclusions and proposals for the next research steps are given.

Keywords: NoSQL, IaaS, home location register, performance, SLA

1 Introduction

Telecommunications operators are used to running their embedded computer systems on proprietary platforms. Typically operators have not shared infrastructures either, but have purchased their own networks. However, this situation is slowly changing. The first step has been taken by the Mobile Virtual Network Operators (MVNO), who have outsourced some part, or even the whole network, to network vendors. MVNOs have also utilized shared radio access networks (RAN) to avoid high initial investment costs. Recently, due to saturated revenues, cost pressures on operability and introduction of flat network architectures, such as Long Term Evolution (LTE), also dominant operators have shown interest in network sharing initiatives.

Moreover, unlike the past, mobile networks are based on commercial computers equipped with the Linux operating system. Mobile software can be easily ported into cloud platforms as such, resulting in shorter integration projects and lower capex costs. Parallel to this, CPU and data storage performance are still developing almost exponentially [4]. This paradigm shift will open novel opportunities for cloud technologies in the telecom sector [7, 23].

It is probable that mobile networks will not change from private and proprietary servers to public clouds in the short term, but telecom networks definitely include areas where cloud options can have a role. Especially mobile application servers and backend support systems might suit cloud computing well. The main drivers for the successful introduction of cloud technologies imply a large variation in traffic patterns or massive data volumes. In addition, telecommunication networks are normally designed based on the peak load, meaning that during off-peak periods systems have a lot of unused capacity. Cloud computing thus offers a natural technology for resource sharing.

However, there are still concerns whether cloud computing meets the carrier grade requirements [15]. Service level agreements (SLA) in areas such as high availability (HA), latency and transactions per second, are strict in several telecom services. One of the most critical mobile network functionalities is the home location register (HLR). HLR is the core element of the mobile system, and for example, is responsible for subscriber authentication and roaming functionalities. HLR also incorporates a risk for single point of failure, resulting in very high SLA requirements.

The paper evaluates, whether cloud technologies can meet the strictest telecom SLA requirements. HLR is used as a use case, although it is clear that the HLR, being the crown jewel of the operator, will not be the first functionality that operators would outsource to the cloud. However, the HLR presents exact SLA requirements, and also benchmark data and tools are available from the existing systems. HLR behavior in the cloud is studied by using two cloud technologies. First of all, all computation is placed into the Infrastructure as a Service (IaaS). IaaS can be applied locally, remotely and in both ways, referring to private, public and hybrid clouds, respectively. Secondly, the HLR benchmark tool, called Telecommunication Application Transaction Processing (TATP), is implemented in the HBase cloud database, which is based on NoSQL technology.

The paper is organized as follows. Firstly, Section 2 presents the background data for the applied technologies. Then, Section 3 describes the measurement setups. The results are shown using in Section 4 with an emphasis on the public cloud environment. The main criteria are latency and transactions per second. Next, the results are discussed, critically reviewed, and also a short business comparison is given in Section 5. Finally, Section 6 summarizes the paper and proposes future research ideas.

2 Background

This section highlights the background of the key cloud computing components, namely IaaS and NoSQL. The chosen NoSQL technology, HBase, and the benchmark tool, TATP, are briefly described.

2.1 IaaS

An IaaS provides the most natural approach for the research. The existing telecom network elements, using the Linux operating system, can be easily ported as such into the IaaS platforms. Compared to the Platform as a Service (PaaS) or Software as a Service (SaaS) alternatives, IaaS offers the best flexibility for its users. Unlike IaaS, PaaS service providers, such as Google App Engine or Microsoft Azure, require that the software is tailored for the associated platform. On the other hand, SaaS provides a complete service that does not allow running your own code. In addition, IaaS supports a large selection of open source software solutions that are compatible with the commercial IaaS market leader, Amazon Elastic Compute Cloud (EC2) [3] By selecting the IaaS approach, the users can avoid the vendor or system lock-in, a feature that is much appreciated by the operators.

From commercial, public IaaS cloud vendors EC2, being a market leader, was a natural choice. On the private cloud side, the selection process was a lot more difficult. There are several alternatives in open source software IaaS platforms. The most well known, EC2 compatible, projects are called Eucalyptus[4], OpenNebula[5] and OpenStack [6]. As one of the more mature projects, Eucalyptus was selected for the private cloud platform, but for future research, OpenNebula and OpenStack are worth closer consideration.

Interoperability and backward compatibility of the software are essential features. Amazon EC2 and Eucalyptus provide an attractive duopoly, where software can be ported with minor efforts from one entity to another. The good interoperability basically enables two different scenarios. First of all, companies may develop their product using their own cloud, and at once a stable phase has been achieved, the software can be commercialized using a public cloud. The second possibility is to utilize a hybrid model, where private and public clouds complement each other, enabling load balancing functionalities. This is a valid scenario also in telecom applications, where the traffic peaks are a common challenge.

2.2 NoSQL

Distributed databases have been at the forefront of cloud computing since the beginning, although the term NoSQL was invented much later. It is an umbrella

[3] `http://aws.amazon.com/ec2/`
[4] `http://open.eucalyptus.com/`
[5] `http://www.opennebula.com/`
[6] `http://openstack.org/`

term for a family of databases that typically do not implement the SQL interface, but are designed scale horizontally to support massive data. Originally the need to create a new kind of database stemmed from the data storage requirements of the first globally scale internet services. Soon, in addition to internal use at social media sites and internet companies, NoSQL solutions became available as services for all developers.

The main differences between a NoSQL and a SQL, i.e. a Relational Database Management System (RDBMS), in a data model are provided interfaces, transaction guarantees and scalability. NoSQL differs fundamentally from the SQL databases that form the basis of telecom database systems. Generally RDBMS is optimal fo online transaction processing (OLTP), and NoSQL for online analytics processing (OLAP) [2]. While a SQL database confirms ACID (atomicity, consistency, isolation, durability) requirements, NoSQL databases typically support BASE (Basically Available, Soft state, Eventually consistent) principles [17].

The modern history of the NoSQL movement as an effort to store web scale data can be seen to have begun in 2003 when Google published details on its Google File System (GFS) [8]. Later in 2006, the company published an article describing Bigtable [5], a distributed storage system built on top of GFS. Imitating Google's efforts, the Apache Software Foundation (ASF) has developed open source clones, called the HBase [3] and Hadoop Distributed File System (HDFS) [19].

HBase and HDFS were chosen for a closer examination due to three reasons. First of all, HBase supports consistent transactions when updating a single row at a time. Secondly, it has a modular design and proven basis, thanks to underlying HDFS and ZooKeeper layers. Thirdly, HBase has active community and support from strong internet companies such as Yahoo. Yahoo has also developed a benchmark tool for cloud storages, including HBase [6].

2.3 HBase

The entire software stack, on which HBase runs, is modeled after the Google Bigtable. An HBase database *table* is divided into *regions* based on row keys. The nodes in an HBase cluster are called *region servers* because they store information in regions. In addition, the cluster has one active *master* and possibly several *backup masters* that will compete to become a master in case the active master goes down. HBase ultimately stores data in HDFS DataNodes and utilizes ZooKeeper for reliable coordination of the distributed system. An HBase master manages region servers by assigning regions to them. On top of the regions, containing actual client data, the system manages the *ROOT* and *META regions*. The ROOT region is assigned first and it has a mapping data to all the META table regions. The META regions in turn keep track of the actual user table regions handled by the region servers. The master registers the server hosting the ROOT region and registers itself to the ZooKeeper quorum. [3]

A ZooKeeper provides a file system-like abstraction for its data tree where the nodes are called *znodes*. The ZooKeeper clients that are usually servers in some distributed system, such as HBase, read and write znodes through a client API. Znodes can be used to store actual payload information, but more often they are used only for storing metadata and configuration data. The ZooKeeper guarantees that all writes to the service are linearizable, and further it guarantees a FIFO order for all requests from a given client. Researchers at Yahoo! evaluated the performance of different sizes of ZooKeeper clusters, loading it with 35 client machines simulating 200 simultaneous clients. As expected from the design decisions made, the reads scale with the size of the cluster, whereas the performance of the writes decreases due to the atomic broadcast. A cluster of 5 servers executed close to 35 000 operations per second under a load of 50 percent reads and 50 percent writes, where each operation consisted of a read or write of 1 KB of data. [12]

HBase stores data in the Hadoop Distributed File System (HDFS), which is consequently a prerequisite for all HBase deployments. Each HDFS cluster has a single master called a NameNode that manages the metadata of the file system. In particular, it keeps track of the mapping of the fileblocks to the DataNodes storing them. The namespace of the file system is of the typical sort with files and directories. In the NameNode, *inodes* that include information about permissions and quotas, represent these files and directories. The inodes and the list of blocks belonging to each file define the state of the NameNode known as the *image*. In normal operation, the HBase master runs on the same server with the NameNode.

2.4 TATP

The Telecommunication Application Transaction Processing (TATP) benchmark aims to measure the performance of a database under load which is typical in telecommunication applications. In particular, it is modelled after the type of queries that are processed in HLR on a GSM network. The benchmark tool is described in detail in the literature [20, 21]. TATP encompasses seven different transactions of which three are reads and four are writes. The description gives probabilities at which each of the transactions is executed in the client. Broadly, 80 percent are reads and 20 percent are writes.

The database industry has been dominated by RDBMSs for several decades, and it still is. Accordingly, TATP benchmark is heavily dependent on SQL, and does not provide functionality to test other kinds of database systems. However, we have taken action and implemented a comparable benchmark for HBase. The schema in TATP consists of four inter-relational tables. When modelling the schema for HBase, the tables were denormalised and finished off with just one table. Denormalisation is a popular approach when designing data models for NoSQL databases.

3 Measurement Setup

This section describes the measurement environment, including a high level view on the mobile architecture and a short description of the benchmark tool transactions. The latter part of the section gives details of the measurement setups.

3.1 Environment

The test environment simulates a real mobile network, where one HLR was loaded by one or several Mobile Switching Centers (MSC). The TATP benchmarking tool emulates the real signalling traffic between the MSCs and HLR. See Fig. 1 for the model. The focus in the measurements was on the SLA, latency and transactions per second. HA measurements were beyond the scope of the research. All measurements were repeated a few times and the diagrams shown are based on average results.

It is noteworthy that the telecom level HA requirement, 99.999 percent, can be achieved by using independent IaaS clusters. For example, utilization of two different Amazon EC2 zones, both with an HA value 99.95 percent, yields together an HA value 99.9999 percent. Similar results can be achieved by using hybrid models.

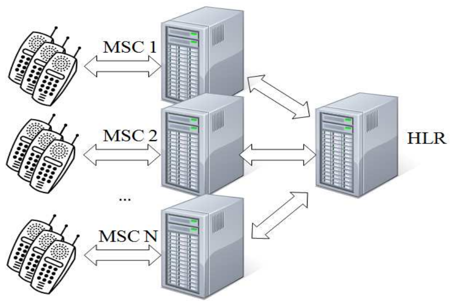

Fig. 1. HLR test environment

3.2 TATP Transactions

The original TATP benchmark database representing HLR includes four different tables called Subscriber, Access Info, Special Facility and Call Forwarding. Subscriber table includes the basic customer data, while Access Info table describes the access type. Special Facility table defines the services subscribers are entitled to, and finally Call Forwarding table reveals forwarding rules. MSC Clients use seven different transactions that either read or write data. The transaction

Table 1. TATP transactions [20].

Transaction name	Type	%	Tables
Get-Subscriber-Data	Read	35	Subscriber
Get-New-Destination	Read	10	Special Facility, Call Forwarding
Get-Access-Data	Read	35	AccessInfo
Update-Subscriber-Data	Write	2	Subscriber, Special Facility
Update-Location	Write	14	Subscriber
Insert-Call-Forwarding	Write	2	Call Forwarding
Delete-Call-Forwarding	Write	2	Call Forwarding

distribution is known from real networks. Table 1 summarizes the transactions, their types, distribution and effected tables. [20].

The TATP version ported for HBase is implemented using Java. It consists of a main program which provides a CLI to the user and separate entities for creating a table, populating it and running the benchmark. The benchmark module spawns a number of client threads according to user input. The client threads execute queries according to the TATP probability distribution and report results back to the benchmark module which writes them to a common file periodically.

3.3 Initial Setup

The rewritten version of TATP was tested in several small HBase clusters. The focus was in transactions per second capability. Running the benchmark is interesting, especially because the results can be compared to existing reports on SQL database performance. One such article [10] reports a throughput of approximately 5500 transactions per second. The performance level was achieved for 200 000 subscribers using carrier grade hardware from the year 2006 and an in-memory database. However, comparing measurement results obtained from different benchmarks testing different databases running on top of different infrastructures head-to-head, is not particularly meaningful. Therefore we use the results in the white papers only to set up a base line so that we know, whether the first results of running HBase in a HLR setting are on the same scale with recent commercial HLR databases.

In the initial measurements the environment was the following. The HBase version 0.20.6 and Hadoop 0.20.2 were run on a multitude of test setups, which all were considerably smaller than what HBase is designed for. Amazon Small EC2 had 1.7 GB memory on a Ubuntu Lucid 10.04 32 bit server. Eucalyptus had also 1.7 GB memory on top of a Ubuntu Lucid 64 bit desktop. Local communication was based on a 100 Mbit/s LAN, and the PCs were equipped with Intel Core 2 Duo processors and 8 GB memory. All setups consisted of a four virtual machine (VM) instance cluster. One instance was a dedicated master running the Hadoop Master Server and HDFS NameNode, two instances were running the

HBase Region Server and HDFS DataNode processes, and the fourth machine was running a single HLR benchmark process and collecting the results.

In the hybrid setup the HBase Region Servers and HDFS DataNodes were split into both EC2 and Eucalyptus, while the HBase Master, HDFS NameNode and benchmark client were running on a local Dell Optiplex 960 desktop. A noteworthy result itself is that we were able to run HBase and HDFS with default settings on Amazon Small EC2 instances without problems.

3.4 Final Setup

The final measurement setup was decided to be based on Amazon EC2 only. It was already beforehand clear that a hybrid IaaS architecture is not optimal for a centralized database system. Furthermore, during the research process it was found that Eucalyptus is not the best platform for a hybrid cloud. The main reason is the lack of necessary management tools for remote instances. In contrast, OpenNebula and OpenStack support these functionalities, and for that reason those IaaS alternatives should be researched more in the hybrid context. On the other hand, private cloud measurement results were mainly limited only by the local hardware applied, having a small difference to the usual HLR environment.

In the later experiments, we investigated the effect of load, replication, database size and node failure on performance by running HBase on a cluster of six Large EC2 instances as shown in Fig. 2. As the characteristics of EC2 instance types are sometimes modified, it is purposeful to specify here that the large instances used in the experimentation were virtual machines with 7.5 GB of memory and two virtual cores with two EC2 compute units, each running a 64-bit Ubuntu Server version 10.04. In addition to one master and four slave nodes, one large instance was hosting the benchmark clients.

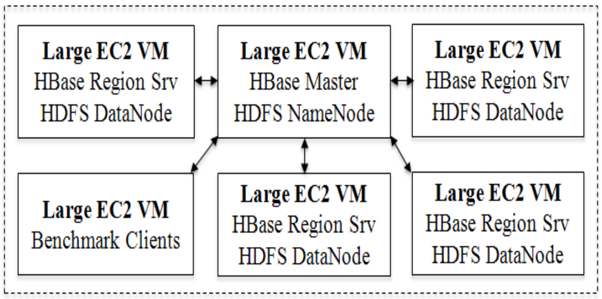

Fig. 2. Final setup

4 Measurement Results

The results section gives the initial measurement result, followed by the main results. The main measurement results, concentrating in the latency and performance factors, are concluded by a brief cost analysis.

4.1 Initial Results

The initial test results are shown in Fig. 3. All transactions per second measurements were made with a database size of 200 000 subscribers. As described above, the comparison of different IaaS platforms with each other is not useful, but on the other hand, the results can be used for getting the big picture of the system. Compared to the four year old carrier grade numbers, the results were encouraging. The Amazon Large EC2 cluster achieved roughly 15 percent performance of the carrier grade system. Even the local, Eucalyptus based IaaS cluster managed to produce reasonable results. Plain workstation and legacy cluster measurements were made to gather experiences of running the benchmark setup in different environments.

The performance of a hybrid setup, consisting of Small Eucalyptus and EC2 instances, was better than with a Small EC2. The results prove that a hybrid IaaS can be made, and that the throughput is roughly the average of the building blocks. The first experiments also revealed that the bottleneck in the measurements was the single benchmark client. In the main measurements this bottleneck was removed by using several parallel clients. In the real networks one HLR is connected to several MSCs, too.

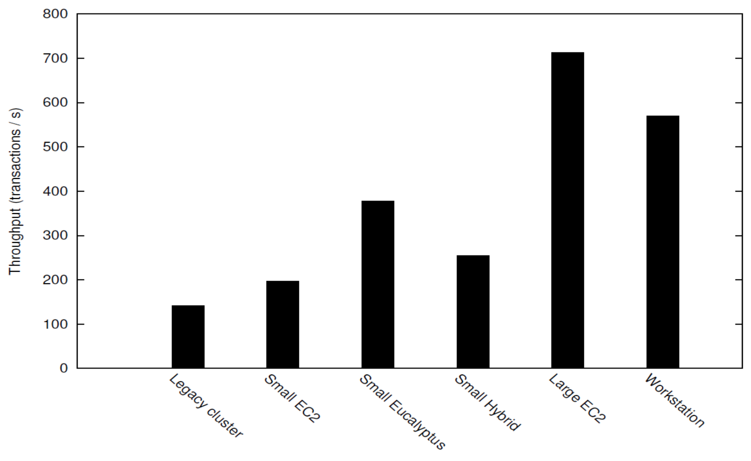

Fig. 3. Initial results

4.2 Main Results

The experiments analyzed here and presented in Fig. 4 and Fig. 5 illustrate a load curve typical for I/O heavy systems. The throughput improves to a certain limit when client processes, i.e. load increases, are added, but after the limit latency grows dramatically. The point of maximal curvature is known as the knee. Our benchmark collects the latency distribution for each of the seven transaction types of TATP separately. Therefore, the response time values shown in the figures are the 95th percentile values perceived by the worst performing client process from the heaviest transaction type.

Replication is a standard way of achieving durability of data in NoSQL databases. Fig. 4 shows the results of the experiment where the goal was to assess the effect of replication on performance. The table was populated with 200 000 subscribers. First of all, the throughput results show that 16 client processes are close to the knee, e.g. a point where the results turn worse. Secondly, we notice that the replication factor does not have a major impact on the throughput. Also the response time holds almost steady independent of the replication factor. We assume that the performance penalty of replication is virtually nonexistent, because even if writes become heavier, reads are scaled across the replicas, which balances the results in a read heavy benchmark.

Fig. 5 presents the effect of the amount of subscribers in the database on performance. The results were gathered when the replication factor was set to three. As expected, the performance gradually decreases as the table size grows. Looking at the results from 16 concurrent client processes, increasing the table size from one to five million subscribers decreases the throughput 32 percent and lengthens the response time 36 percent.

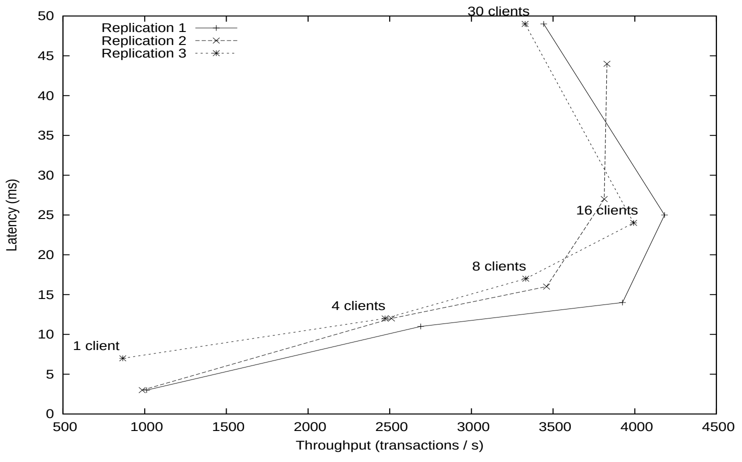

Fig. 4. Impact of replication factor and number of clients

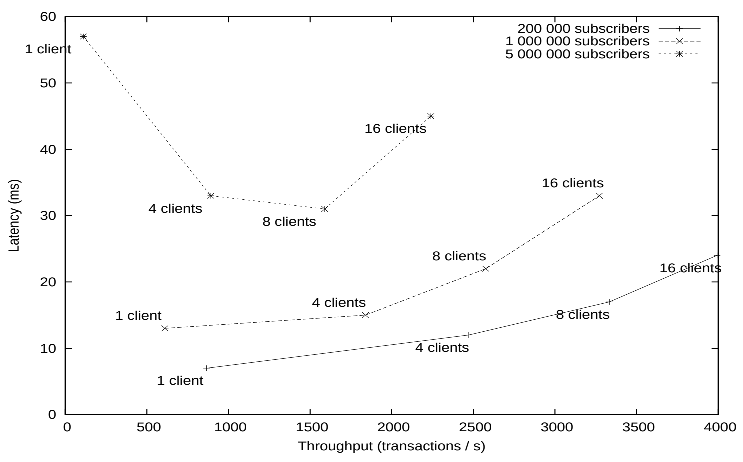

Fig. 5. Impact of database size and number of clients

To verify that the HDFS replication gives protection from node failures, we studied the effect of killing one slave node in the middle of a benchmark run. The measurement was done with a database size of 1 million subscribers and replication factor two. Fig. 6 shows the effect of one failing node 10 seconds after the launch of the run as perceived by four client processes. The throughput values are gathered once a second for each client and the results are stacked in the presentation. In this sample the distributed database quickly recovers from the failure and continues serving clients within two seconds. The perceived recovery time in the experiment would be too much for real-time telecommunications applications, but it could be improved by tuning the parameters related to timeout mechanisms.

4.3 Results vs. Requirements

In order to give an idea of the load generated by the modified version of TATP, the performance testing tool bundled with HBase was also run on the test setup of six large EC2 instances. We run the performance test on the master node using 16 client threads and disabled MapReduce for it. By default, the test populates a table with one million rows of 1 kB each. In our experiment the random Read test took 1492 seconds, which leads to a throughput of 5.36 Mbit/s per client thread, and an aggregated throughput of 85.8 Mbit/s.

The 3GPP has defined the HLR performance requirements in their general specification [1]. According to that each subscriber produces on average 1.8 mobility related and 0.4 call handling related transactions per hour. Together this yields 2.2 transactions per hour per subscriber. With this information and the total number of subscribers, a requirement curve for transactions per second can be defined. Pulling together the requirements for HLR performance and the

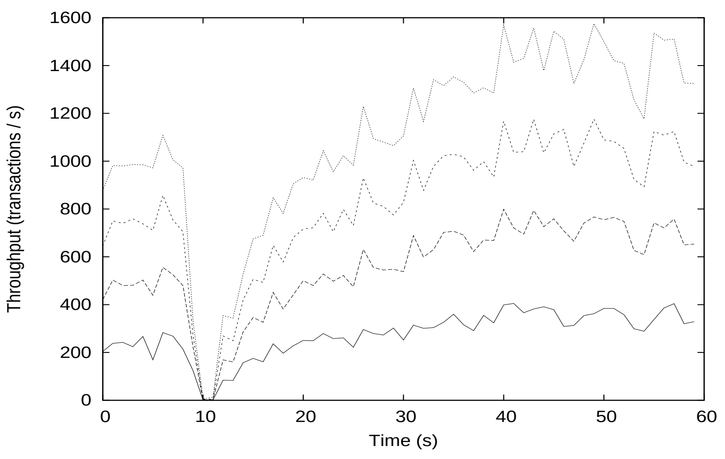

Fig. 6. Recovery time from node failure

measurements from the HBase benchmark leads us to the conclusion that up to 4 million subscribers could in theory be supported by six large EC2 instances. Measurement results and comparable requirements are shown in Fig. 7.

In the experiment each subscriber added 3.7 kB to the database size leading to a total of 18.5 GB for 5 million subscribers. This is still in the area that can be handled in main memory, and therefore existing HLR solutions can support such deployments using an in-memory database. Similarly to most NoSQL databases, HBase does not support transactions, which span multiple rows, but on the other hand HBase guarantees that a single row remains consistent at all times. In an HLR all transactions read or update a single subscriber, and therefore the database was modelled so that all data related to a single subscriber is on one row.

4.4 Cost Evaluation

Following the ideas presented by Walker [22], we propose a simple model to help decide whether to run a database in the cloud or in a hosted rack space. A third alternative would be to place the servers in private premises, but in that case the cost structure becomes very complex, because costs related to factors such as the property, security, UPS, generators, etc. should be taken into account. Entire studies have been made in this area and one of the most insightful ones is written by Greenberg et al. [9]. However that article focuses on a large data center, whereas our case example is an individual application running on a relatively small cluster. Therefore, we simply take the cost of hosted rack space, server hardware and disk replacements as the basis for our formula. On the other hand, the Equation 1 can be applied to servers within private premises only if the $\alpha_{hosting}$ is modified to account for the above mentioned infrastructure costs.

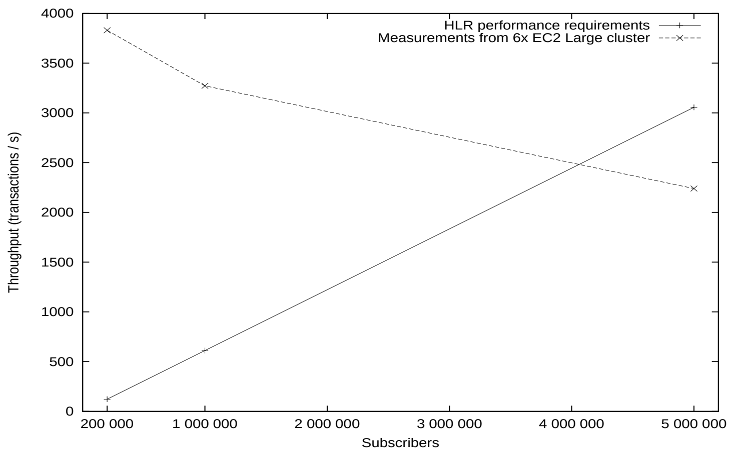

Fig. 7. Results vs. requirements

$$C_m = n_{servers} * (\alpha_{hosting} + \beta_{hardware} + \gamma_{disks}) \tag{1}$$

$$beta_{hardware} = p * \frac{\frac{i}{12}(1 + \frac{i}{12})^m}{(1 + \frac{i}{12})^m - 1} \tag{2}$$

$$\gamma_{disks} = \Psi \frac{R}{12} * \epsilon \tag{3}$$

In Fig. 8, we present the results calculated from using the Equation 1 compared with current EC2 pricing. The hypothetical cluster has six nodes and, in the case of hosting it in a rented rack, each node costs 2000 euro, while a rack unit costs 80 euro/month including the internet connection. The factor γ_{disks} in

Table 2. Parameters in the formula presented in Equations 1-3

Parameter	**Unit**	**Description**
C_m	euro	Monthly cost of running servers in hosted environment
$n_{servers}$		Number of servers
$\alpha_{hosting}$	euro	Monthly cost of hosted rack space per server (1U)
$\beta_{hardware}$	euro	Amortized monthly cost of a server
i		Interest rate
p	euro	Price of a server
m	months	Amortization period in months
γ_{disks}	euro	Monthly cost of replacement hard disks
Ψ		Quantity of hard disks per server
R		Annual replacement rate (ARR)
ϵ	euro	Price of a hard disk

Equation 1 adds in the cost of replacing failed hard disks similarly to Walker's model. The extensive research of 100 000 hard disks carried out by Schroeder and Gibson [18] discovered that a realistic annual replacement rate (ARR) is around 3 percent and therefore we use the value for R in the calculation. However, even if a reasonable service fee for the disk replacements is included, γ_{disks} is not a significant cost factor in this model. The price of the servers $\beta_{hardware}$ is amortized using the standard annuity formula and an interest rate of 5 percent over a period of time. It is shown on the x-axis.

Amazon EC2 provides two alternative payment options: simple hourly costs per server or a one-time fee and reduced hourly costs. Amazon calls the latter a reserved instance purchasing option. Fig. 8 presents the data points for the costs of reserved instance plans for either a period of 12 or 36 months. The one-time fee amortized for the respective period in the same fashion as $\beta_{hardware}$ and the hourly costs are simply multiplied by the number of hours in a month. All prices were converted from US dollars to euros using a coefficient of 0.75.

Interestingly, monthly cost of running a Double XL instance with 34.2 GB of memory is 700 euro with a three year reserved instance contract, whereas a comparable carrier grade server such as Oracle Netra X4270 could be purchased for 8000 euro, which yields a monthly amortization cost of 240 euro. Comparison of the specification and pricing of EC2 instances with S3 storage service yields the result that storing data on the hard disk of an instance instead of S3 may be smart in some cases. On the other hand, S3 provides many features that an administrator would have to add in the case of using an instance, and hence S3 can be regarded as having a higher value service.

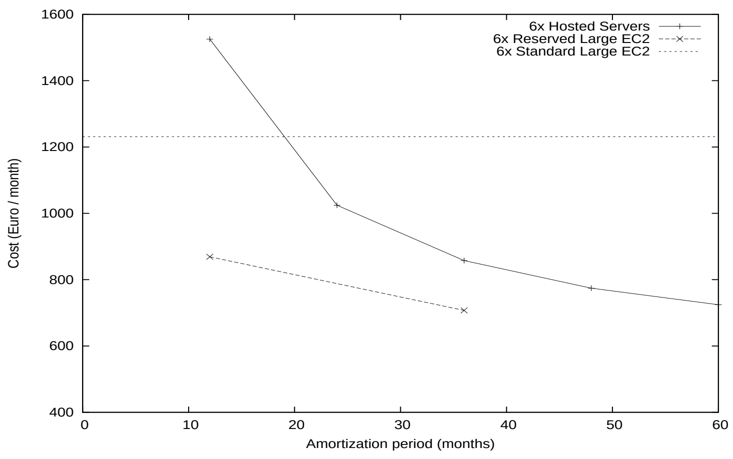

Fig. 8. Cost evaluation

5 Discussion

Cloud computing offers a new perspective on mobile network optimization. Although the HLR is not a primary candidate for the cloud, the results give evidence that some other mobile network elements could be placed there. Application servers, such as SMS Center (SMSC), IP Multimedia Subsystem (IMS) and Service Delivery Platform (SDP), are examples of those. Backend processes can provide an even better solution area [11]. Billing, customer care and maintenance systems create a lot of data that could be computed by cloud infrastructures. A general purpose cloud can also be provided by mobile network vendors, who might use their large customer base to benefit from the statistical multiplexing. The same approach can work with operators, who operate in several countries and continentals. It can be expected that in future mobile networks, such as Long Term Evolution (LTE), operators will compete and cooperate at the same time, leading to network sharing initiatives.

The HLR benchmarking measurements produced a lot of information about the cloud computing opportunities in the telecom sector. The main lesson is that even the strict telecom SLA requirements can be achieved with both public and private clouds. The initial measurements also revealed that to match the existing RDBMS solutions, NoSQL databases have to fully utilize horizontal scalability. In addition, configuration parameters must be properly tuned, enterprise class infrastructure must be used and several client processes must be deployed.

IaaS, both private and public versions, operated according to expectations in the measurements. Due to the short history of private clouds, they are still developing. Amazon EC2, on the other hand, is already a mature product. The hybrid cloud was a side track in this research. It became evident that the hybrid cloud does not suit well to a centralized database system. In addition, the hybrid setup must be carefully designed to overcome configuration, management and load balancing challenges. For certain applications a hybrid cloud can be an interesting option to optimize the dimensioning for peak loads [14]. However, the database solutions should be centrally located backed by 2N or N+1 redundancy algorithms. Database distribution will increase latency times and create unnecessary functional complications as well. For example, security and regulation challenges would become high. On the other hand, a public cloud can successfully host voice applications [23].

A brief financial comparison between Amazon EC2 and hosted rack space is given. As a conclusion of the exercise the setup of EC2 instances reserved for a three year period is the most attractive solution cost-wise, whereas the hosted alternative is more cost-efficient than standard EC2 instances starting from an amortization period of approximately 18 months. In the high-end, there is a fair premium in Amazon prices. When it comes to I/O performance a physical 2000 euro server is very likely to outperform a Large EC2 virtual instance. It is also worth mentioning that the EC2 pricing structure is the most versatile, including also spot prices [13]. Secondly, unlike in clouds [9, 22], the weight of computing power and storage is marginal in the HLR price formula. However, in the application servers computing costs are becoming ever more dominant.

6 Conclusions

We have introduced research on how cloud computing performance meets the SLA requirements of mobile networks. The home location register (HLR) was chosen as an example for benchmarking measurements. The HLR benchmark tool, originally developed for the SQL databases, was ported into the NoSQL, HBase specific environment. The software instances were deployed on private, public and hybrid Infrastructure as a Service (IaaS) platform. The measurement results indicate that cloud technologies can achieve the requirements of mobile network latency and transactions per second. Also telecom high availability (HA) targets can be met by using parallel computing zones. It is recommended that future studies should evaluate whether cloud technologies can be applied to mobile application servers and backend processes. Also Long Term Evolution (LTE) will provide interesting research opportunities on network sharing between operators. Finally, hybrid clouds deserve attention in managing traffic peaks.

Acknowledgments. The work is supported by Tekes (the Finnish Funding Agency for Technology and Innovation, www.tekes.fi) as a part of the Cloud Software Program (www.cloudsoftwareprogram.org) of Tivit (Strategic Centre for Science, Technology and Innovation in the Field of ICT, www.tivit.fi).

References

1. 3GPP: Technical performance objectives TS 43.005 v9.0.0 (2010)
2. Abadi, D.J.: Data management in the cloud: limitations and opportunities. IEEE Data Engineering Bulletin, 32(1) (2009)
3. The Apache Software Foundation. HBase, `http://hbase.apache.org/`. (2011) Accessed 29 Aug 2011
4. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R.H., Konwinski, A., Lee, G., Patterson, D.A., Rabkin, A., Stoica, I.: Above the Clouds: A Berkeley view of cloud computing. Technical Report no. UCB/EECS-2009-28, Electrical Engineering and Computer Sciences, University of California at Berkeley (2009)
5. Chang, F., Dean, J., Ghemawat, S., Hsieh W.C., Wallach, D.A., Burrows, M., Chandra, T., Fikes, A., Gruber, R.E.: Bigtable: a distributed storage system for structured data. In: 7th Symposium on Operating Systems Design and Implementation (OSDI 06), Seatle, WA, USA, 6-8 November 2006, pp. 205–218. USENIX Association (2006)
6. Cooper, B.F., Silberstein, A., Tam, E., Ramakrishnan, R., Sears, R.: Benchmarking cloud serving systems with YCSB. In: Proceedings of 1st ACM Symposium on Cloud Computing (SoCC '10), Indianapolis, Indiana, USA, 10-11 June 2010 (2010)
7. Gabrielsson, J., Hubertsson, O., Más, I., Skog, R.: Cloud computing in telecommunications. Ericsson Review, vol. 1, pp. 29–33 (2010)
8. Ghemawat, S., Gobioff, H., Leung, S.-T.: The Google file system. In: 19th Symposium on Operating Systems Principles (SOSP03), New York, NY, USA, 19-22 October 2003. ACM (2003)
9. Greenberg, A., Hamilton, J., Maltz, D.A., Patel, P.: The Cost of a Cloud: Research Problems in Data Center Networks. ACM SIGCOMM Computer Communication Review, Vol. 39, no. 1 (2009)

10. Gupta, N.: Enabling High Performance HLR Solutions. Netra Systems and Networking, Sun Microsystems Inc., version 2, 17 April 2006 (2006)
11. Hajjat, M., Sun, X., Sung, Y.-W.E., Maltz, D., Rao, S., Sripanidkulchai, K., Tawarmalani, M.: Cloudward bound: planning for benefiticial migration of enterprise applications to the cloud. In: SIGCOMM, New Delhi, India, 30 August-3 September 2010. ACM (2010)
12. Hunt, P., Konar, M., JUnqueira, F.P., Reed, B.: Zookeeper: Wait-free coordination for Internet-scale systems. In: the 2010 USENIX Annual Technical Conference (USENIXATC'10), Boston, MA, USA, 23-25 June 2010. USENIX Association (2010).
13. Mattess, M., Vecchiola, C., Buyya, R.: Managing peak loads by leasing cloud infrastructure services from a spot market. In: the 12th IEEE International Conference on High Performance Computing and Communications (IEEE HPCC 2010), Melbourne, Australia, 1-3 September 2010 (2010)
14. Moreno-Vozmediano, R., Montero, R.S., Llorente, I.M.: Elastic management of cluster-based services in the cloud. In: the 1st Workshop on Automated Control for Datacenters and Clouds (ACDC '09), Barcelona, Spain, 19 June 2009 (2009)
15. Murphy, M.: Telco cloud. Presentation at Cloud Asia, Singapore, 5 May 2010 (2010)
16. Nurmi, D., Wolski, R., Grzegorczyk, C.: The Eucalyptus open-source cloud-computing system. In: the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2009), Shanghai, China, 18-21 May 2009 (2009)
17. Pritchett, D.: Base: an acid alternative. ACM Queue, vol 6, no. 3, pp. 48–55 (2008)
18. Schroeder, B., Gibson G.A.: Disk failures in the real world: What does an MTTF of 1,000,000 hours mean to you? In: 5th USENIX Conference on File and Storage Technologies (FAST'07), San Jose, CA, USA, 14-16 February 14-16 2007. USENIX Association (2007)
19. Shvachko, K., Kuang, H., Radia, S., Chansler, R.: The Hadoop Distributed File System. In: 26th Symposium on Mass Storage Systems and Technologies (MSST), Incline Village, NV, USA, 3-7 May 2010. IEEE (2010)
20. Strandell, T.: Open source database systems: systems study, performance and scalability. Masters thesis, University of Helsinki, Faculty of Science, Department of Computer Science, and Nokia Research Center (2003)
21. TATP. Telecom Application Transaction Processing Benchmark, `http://tatpbenchmark.sourceforge.net/`. (2011) Accessed 29 Aug 2011
22. Walker, E., Brisken, W., Romney, J.: To lease or not to lease from storage clouds. Computer, April 2010, pp. 44-50. IEEE (2010)
23. Venugopal, S., Li H., Ray P.: Auto-scaling Emergency Call Centres using Cloud Resources to Handle Disasters. In: 19th International Workshop on Quality of Service (IWQoS), San Jose, CA, USA, 6-7 June 2011 (2011)